WP 7.2 Project Deliverable

Journal articles

Project Number	IST-2000-29266
Project Title	Virtual Real Time Fire Emergency Simulator
Deliverable Type	Report
Deliverable Class	Internal

Deliverable Number	D7.2
Title of Deliverable	Dissemination through journals.
Nature of the Deliverable	Report
Contributing WPs	WP 7
Contractual Date of Delivery	28 February 2004
Actual Date of Delivery	30 August 2004
URL	www.virtualfires.org
Authors	Sascha Schneider (FIGD), Michael Schmidt (FIGD), Volker Luckas (FIGD), Gunther Lenz (SiTu), Thomas Reichl (SiTu), Wilhelm Brandstätter (CD), Christian Redl (CD), Gert Svensson (KTH), Kai-Mikael Jää-Aro (KTH), Alain Bochon (Alpetunnel), René-Michel Faure (CETU), Klaus Schäfer (FDDo), Rainer Koch (FDDo), José Luis Ajuria (EUVE), Christian Redl (CD), Redactor : R.M. Faure
Contact Details	Institute for Structural Analysis / SiTu Research Univ. Prof. Dipl.-Ing. Dr. techn. Gernot Beer Lessingstrasse 25/II 8010 Graz / Austria Tel.: +43 316 8736180 Fax: +43 316 8736185 Email: gernot.beer@ifb.tu-graz.ac.at

Abstract	List of documents shown in journals.
Keywords	Papers, documents, news.

Contents

1 Overview

This report lists all papers that have been published in journals during the project.

2 Published papers

2.1 CD

2.1.1 Virtual Fires - Experiences with developing a LBGK based Real Time Tunnel Fire Simulator for Virtual Environments

Brandstätter, W., Redl, C.,

Christian Doppler Laboratory for Applied Computational Thermofluiddynamics

LNCS 2657, p 1062 - 1071 (2003)

2.2 FIGD

2.2.1 Rendering Large (Volume) Datasets: A new Parallel Visualization System

Sascha Schneider, Thorsten May, Michael Schmidt

Fraunhofer Institut für Graphische Datenverarbeitung

Journal of WSCG, 11(3):418–424, February 2003 (Pilzen)

3 Appendix: Full Papers

VIRTUAL FIRES
Expericences with developing a LBGK based real time tunnel fire simulator for virtual environments

Wilhelm Brandstätter and Christian Redl

Christian-Doppler-Laboratory for Applied Computational Thermofluiddynamics,
Mining University Leoben,
Franz-Josef-Strasse 18, 8700 Leoben, Austria

Abstract. In the VIRTUAL FIRES project a LBGK based simulator is coupled to a virtual reality environment for a interactive real time simulation of tunnel fires. The simulator not only has to deal with turbulent buoyant flows but it must also be able to calculate a big number of timesteps each second for the real time visualisation. The general concept as well as some implementation details are highlighted. Furthermore first results fire simulation in a real scale tunnel containing two trucks are presented.

1 Introduction

Recent fire accidents in tunnels (Mont Blanc tunnel, Tauern tunnel) [1] have raised attention on fire prevention and fire fighting means. Real fire tests in tunnels are very costly and are environmentally not friendly due to the production of toxic smoke. Therefore they can only be carried out occasionally. In addition fire fighters are endangered if they exercise with real fires.

All these aspects leaded to the idea of a virtual tunnel fire. Therefore a simulator is developed and connected to a virtual environment which allows a realistic visualisation of smoke and flame spread.

In this paper the authors want to describe their experiences in developing an interactive real time tunnel fire simulator for virtual environments based on a Lattice BGK model [2]. The reasons for selection a Lattice Boltzmann based model for simulation a large scale problem are explained. Furthermore first results obtained with the simulation program are presented. The VIRTUAL FIRES simulator is an example how to successfully apply the Lattice BGK method to real world large scale problems. Detailed information about the project and the consortium can be found at the projects web page [3].

2 General system layout

The VIRTUAL FIRES simulator consists of three main parts :

- A database contains all information about the tunnel (geometry, ventilation system, ...) and data concerning the fire representation. Additionally the results of the simulation are stored in the database.
- A Lattice BGK based simulator performs the calculation of turbulent temperature and smoke spread.
- The results are displayed in a virtual reality (VR) environment (CAVE, HMD) using algorithms for a realistic visualisation of smoke.

The VIRTUAL FIRES user is able to retrieve all information from the database from within the VR environment. A rough model of the tunnel is displayed in the visualisation device and the user can place an arbitrary number of vehicles in the computational domain. For each object which should be set on fire information about energy and smoke release are obtained from the database.

During the simulation the results are sent to the visualisation engine upon request. Ideally this would be 25 times per second. The user is able switch on and off the ventilation device of the tunnel interactively from within the VR environment and observe the impact on the smoke spread.

The results used for visualisation are usually not stored due to memory constrains. However, the results are stored frequently in the database allowing the user to restart the simulation at certain points in time using different ventilation settings.

3 Numerical model

3.1 Reason for selecting a Lattice BGK model

Although there are several drawbacks of the LBGK model there are also some strong reasons to select it.

- The regular grid structure and explicit nature allows a very efficient parallelisation. The parallel version scales almost linearly up to a large number of processors.
- The grid generation can be done automatically within the simulator. This is especially important as the potential users do not have much experience in grid generation.
- The regular structure of the computational grid is similar to the grid used for visualisation. The interpolation process from the grid used in the simulation and the one used for visualisation is straightforward.
- As a large number of simulation results are required each second for a real time visualisation explicit methods are suitable.

The authors want to stress that the decision for using a Lattice Boltzmann method instead of a more conventional finite volume formulation is also based on the innovative aspect of applying a LGBK model to large scale problems.

3.2 Turbulent Lattice BGK model

Currently two implementations of the Lattice BGK model are under investigation. The first one is a standard incompressible formulation of the Lattice Boltzmann equation in combination with thermally induced body forces.

The second LBGK model used in the VIRTUAL FIRES project is based on the low Mach number formulation of Filippova and Hänel [4] extended to account for buoyancy effects (as done for laminar jet diffusion flames by Lee et al [5]).

For both models turbulence effects are taken into account by using a Smagorinsky type sub grid scale model including turbulence production due to buoyancy [6]. The spread of smoke can additionally be modelled using an additional distribution function for the smoke concentration.

In this paper only the incompressible formulation is decribed as investigations using the variable density model are in progress.

The incompressible Lattice Boltzmann equation can be written as :

$$p_i(t+1, \mathbf{x}+\mathbf{e}_i) = p_i(t, \mathbf{x}) - \frac{1}{\tau}(p_i(t, \mathbf{x}) - {p_i}^e q(t, \mathbf{x})) \tag{1}$$

where the equilibrium distributions are given by

$${p_i}^{eq} = t_i\{p + p_0(\frac{\mathbf{e}_{i\alpha}\mathbf{u}_\alpha}{{c_s}^2} + \frac{\mathbf{u}_\alpha\mathbf{u}_\beta}{2{c_s}^2}(\frac{\mathbf{e}_{i\alpha}\mathbf{e}_{i\beta}}{{c_s}^2} - \delta_{\alpha\beta}))\} + S_i \tag{2}$$

The energy equation is treated as a scalar transport equation for the temperature within the LBGK framework :

$$g_i(t+1, \mathbf{x}+\mathbf{e}_i) = g_i(t, \mathbf{x}) - \frac{1}{\tau}(g_i(t, \mathbf{x}) - {g_i}^e q(t, \mathbf{x})) + S_{T,i} \tag{3}$$

with the equilibrium distributions given by

$${g_i}^{eq} = t_i T\{1 + (\frac{\mathbf{e}_{i\alpha}\mathbf{u}_\alpha}{{c_s}^2} + \frac{\mathbf{u}_\alpha\mathbf{u}_\beta}{2{c_s}^2}(\frac{\mathbf{e}_{i\alpha}\mathbf{e}_{i\beta}}{{c_s}^2} - \delta_{\alpha\beta}))\} \tag{4}$$

In the above equations p_i and g_i are the distribution functions, $\mathbf{e}_i$ are the lattice vectors and S_i and $S_{T,i}$ are the corresponding source terms. The source term to the transport equation for the temperature is of course only added within the fire region. The lattice dependent weighting factors for the D3Q19 model are $t_0 = \frac{1}{3}$ for the rest particle distribution, $t_{1-6} = \frac{1}{18}$ for the cartesion directions and $t_{7-18} = \frac{1}{36}$ for the diagonal directed lattice vectors. The lattice speed of sound is $c_s = \frac{1}{\sqrt{3}}$.

The hydrodynamice quantities can be obtained by taking the moments of the particle distribution functions :

$$\rho = \sum_i p_i \tag{5}$$

$$\rho\mathbf{u} = \sum_i p_i\mathbf{e}_i \tag{6}$$

$$T = \sum_i g_i \tag{7}$$

The turbulent relaxation parameter τ_ν and τ_D are calculated as follows :

$$\tau_\nu = \frac{6(\nu + \nu_t) + 1}{2} \tag{8}$$

$$\tau_D = \frac{6(D + D_t) + 1}{2} \tag{9}$$

The turbulent transport coefficients are calculated by using a Smagorinsky sub grid scale model :

$$\nu_t = C_s \Delta \sqrt{(2\bar{S_{ij}}\bar{S_{ij}} - \frac{1}{Pr_t}\frac{\partial T}{\partial x_g})} \tag{10}$$

$$D_t = \frac{\nu_t}{Pr_t} \tag{11}$$

In the above equations C_s is the Smagorinsky constant, Δ is the filter width, which is equal to the lattice spacing and Pr_t is presumed turbulent Prandtl number. x_g is the direction of the gravity vector.

Since the density differences away from the fire source are relatively small the Boussinesq approximation is assumed to be valid. The resulting source term $S = -\beta g(T - T_{ref})$ is added to the equilibrium distribution functions.

A comparison between the results obtained with both LBGK models and a detailed decription of the variable density implementation are presented in a forthcoming paper [7].

3.3 Representation of the fire

For the fire modelling a volumetric heat source model is used [8]. A fire source region is predefined and source terms of smoke and energy are added to the evolution equations within this region. As for many engineering applications the details of the combustion process are not important for tunnel fires. Usually the combustible material is not known and the reactions are far too complex to be considered in detail. For the design of tunnel ventilation systems empirical smoke and energy release curves are used. These curves are available for a number of scenarios (e. g. including different types of vehicles like cars or HGVs) and show the smoke and heat release over time. A typical curve [9] used for the VIRTUAL FIRES project is shown in fig 1.

Radiation heat transfer is included using the radiative fraction approach described in [10]. In this approach the thermal radiation in the fluid is ignored and a fixed fraction (typically between 0.2 and 0.4)of the total heat release rate is assumed to be lost without affecting the temperature field.

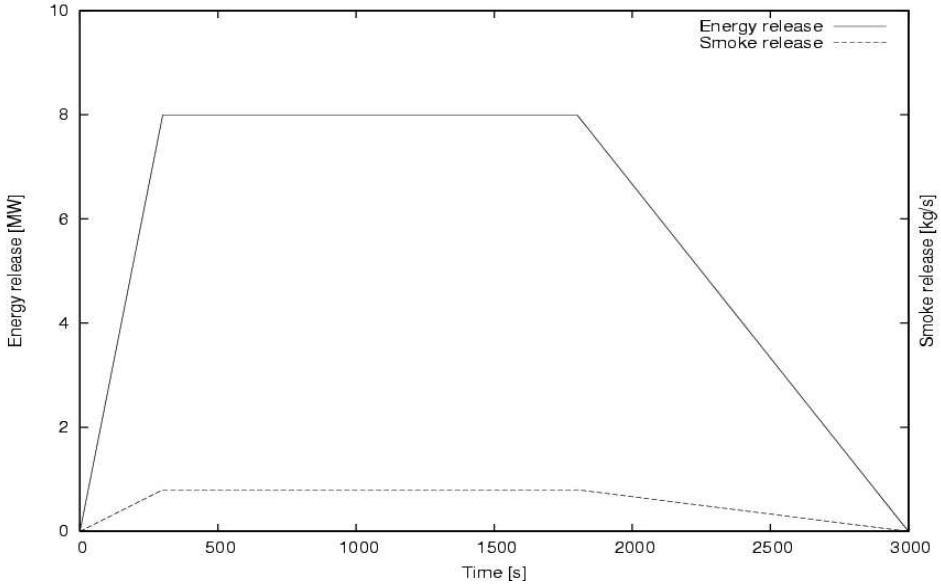

Fig. 1. Energy and smoke release curve for a small heavy goods vehicle [9]

3.4 Multidimensional coupling

To extend the computational domain the 3D model is coupled to a one dimensional two zonal model on both sides. The two zonal model represents the flow field in the tunnel as one upper hot zone below the ceiling and one lower cold zone near the floor [11]. The coupling between the 3D and 1D model along the interface follows the idea described in [12] and is done as follows : The three dimensional cross section is subdivided into two areas. The hydrodynamic variables are averaged across both areas using an averaging procedure which accentuates the spatial centre. These averaged values are used as boundary conditions for the 1D sequence which on the other hand provides the boundary values for the 3D section. The extension of the areas are dynamically adjusted during the simulation. The one-dimensional transport equations are solved by an simple finite difference scheme. Fig. **??** shows an outline of the multi dimensional coupling and the domain decomposition for parallel computing.

4 Parallelisation and Communication

4.1 Parallelisation

The parallelisation can be done very efficiently using the MPI library. To achieve the maximum performance the computational domain is currently restriced to

a rectangular shape. The domain decomposition is done either using a one-dimensional coordinate decomposition or a multi-dimensional coordinate bisection method. Prior to the domain decomposition the expected surface/volume ratios for both methods are roughly estimated and the simulator automatically selects the algorithm which is best suited for the available number of processors.

4.2 Communication

The communication between virtual environment and the simulation program is done via Unix sockets using the communication libraries provided by the visualisation program [13]. For output of the results three options were considered:

- Each processor writes its data to a file belonging to the processor. The data are subsequently gathered within the visualisation part.
- Each processor writes its data to one common output file.
- One processor gathers all data from all other processors and writes it to one output file.

It was decided to use the third option where only the master processor is concerned with the data transfer. It allows a better separation of the simulation and visualisation part and not every processor needs to have access to a writeable file system mounted. Furthermore it is possible to run the simulation on a remote host.

4.3 Performance

The simulation program aims at delivering results of 25 time steps per second for the visualisation. Due to the explicit nature of the Lattice Boltzmann method the computational time step may be smaller than the one needed for visualisation. Preliminary simulations show that currently about 50000 cells can be calculated in real time using the available computing facilities at the Parallel Data Center at KTH Stockholm [14]. The authors found this parallel performance remarkable as due to the real time requirements the computational domains for each processor are rather small. Additionally the gathering of results on the processor responsible for the I/O can take as long as ten times the calculation of one time step.

5 Results

5.1 Testcase : Tunnelfire

As an examplary testcase the tunnelfire experiment by Xue et al [8] is presented. pointed out that none of the combustion model used in their study are able to predict velocity and temperature fields in all test cases equally well.

The experimental set up is as follows :

The tunnel section is 6 [m] long with a rectangular cross section of 0.3 [m] height and 0.9 [m] width. The fire source of 3.15 [kW] is located 1.5 [m] from the inlet and occupies an volume of 0.18 x 0.15 x 0.01 [m3]. The ventilation velocity is about 0.13 [m / s], ambient air temperature is 300 [K]. In contrast to the numerical study in the original paper the heat loss to the tunnel walls is neglected and the wall temperature is set to 300 [K].

The temperature distributions are reported at distances 0.9 [m], 3.3 [m] and 5.1 [m] from the inlet.

Xue et al [8] pointed out that none of the combustion model used in their study are able to predict velocity and temperature fields in all test cases equally well.

Due to the limitations of the volumetric heat source model it is not expected to achive very good quantitative agreement with the experimental data. Nevertheless the general flow patterns are resolved relatively well by the Lattice BGK model :

Despite the longitudinal ventilation there is some backflow against the ventilation direction. A strong stratification of hot gases beneath the tunnel ceiling occurs.The time averaged temperature profiles obtained by the Virtual Fires simulator are in reasonable agreement with the ones obtained by Xue et al [8] using finite volume method in combination with a volumetric heat source model. Nevertheless both methods perform relatively poor in predicting the measured temperatures at locations near the fire. This is mainly due to the crude representation of the fire in the volumetric heat source model. An interesting observation is that the Boussinesq approximation performs surprisingly well for this testcase. As the main point of interest of the Virtual Fires project is to study the spreading of hot gases and not the combustion process itself it is justified to use this approximation at least in the first stage.

5.2 Testcase : Burning Truck within a tunnel

The results of a typical simulation run using the VIRTUAL FIRES simulator are presented below.

The configuration shown in fig. 2 consists of a section of a tunnel containing two exhaust air outlets located at the ceiling. This corresponds to a usual installation in a modern full cross ventilated tunnel [15]. A certain underpressure is assigned to the extraction openings in order to model the removal of hot gases. The effects of natural ventilation are taken into account by applying a pressure difference along the tunnel axis. In reality this longitudinal velocity is induced by differences in altitude of the two tunnel portals or wind outside the tunnel.

Two heavy goods vehicles (HGV) are placed within the tunnel. One trailer is defined as fire zone and set on fire. The smoke and energy release curve shown in fig. 1 is used.

Fig. 3 - 5 depict the results of two simulated scenarios in a relatively early stage after the fire onset. All figures show instantaneous temperture isosurfaces. The two scenarios are as follows:
In the first one the ventilation system is acting in normal mode and natural

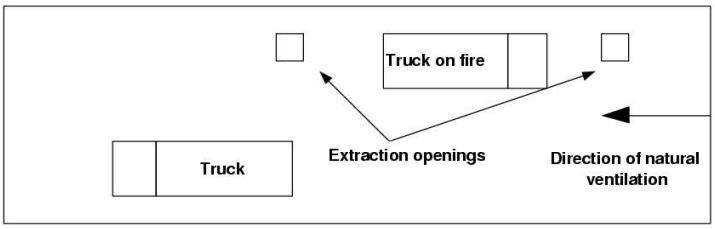

Fig. 2. Outline of the simulated scenario

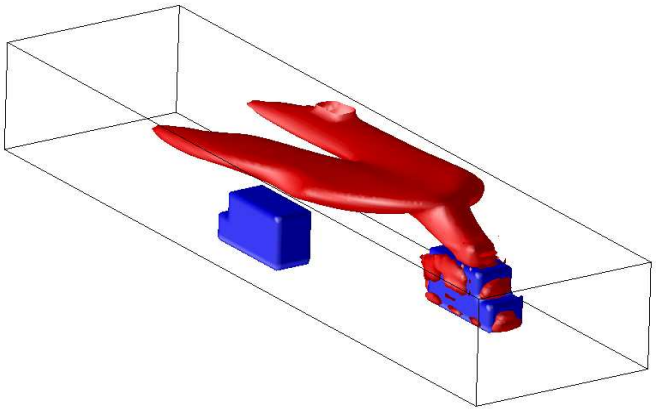

Fig. 3. Ventilation system acting in normal mode. Temperature = 423 [K]

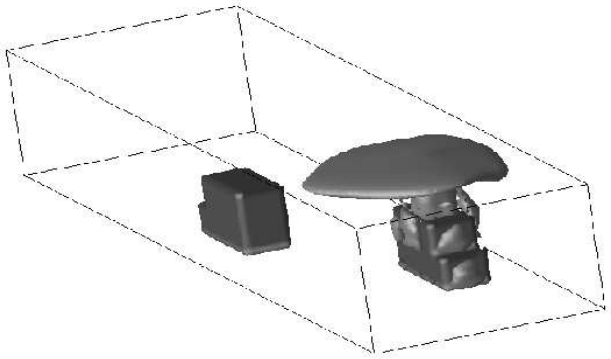

Fig. 4. Ventilation system acting in emergency mode. Temperature = 423 [K]

ventilation is acting. As expected the hot gases are driven in direction of the natural ventilation. The extraction opening located in front of the burning truck has minor influence on the spread of smoke and the one located behind the truck is able to extract only a small fraction of hot gases. In fig. 3 the 423 [K] surface is depicted.

In the second scenario represents the emergency mode. The ventilation system is working at full extraction capacity. As can be seen from fig. 4 a big fraction of hot gases are extracted by the ventilation opening in front of the burning truck. The zone of hot gases is very narrow compared to the first scenario. Fig. 5 shows the relatively low temperature zone at 323 [K].

The computational domain contains about 100 000 cells. Despite the rough resolution the results are in reasonable agreement compared to results obtained using a commercial CFD program with standard $k - \epsilon$-model [16].

6 Conclusion

The selection of a LBGK method working on an uniform grid enables the simulation of complex thermo-fluiddynamic problems such as tunnel fires. The presented results are very promising and the authors hope that the VIRTUAL FIRES simulatior will assist fire fighters and tunnel operator in dealing with fire hazards in tunnels. Future developments will focus on the improvement of interaction between virtual environment and simulation. It is planned to include the possibility of using fire nozzles which can be steered by the user interactively.

Acknowledgements

The authors want to express their graditude to the European Commission for financing the EU project VIRTUAL FIRES (IST-2000-29266) and to all project partners.

References

1. Hörhan, R. : Tunnel Accidents and their Impact on Relevant Guidlines in Austria. In: Pischinger, R.(ed.): Proceedings Int. Conf. Tunnel Safety and Ventilation. VKM-THD Mitteilungen Vol. **80**. Verlag der Technischen Universität Graz, Graz(2002), 1–8.
2. Chen, S., Doolen, G. D. : Lattice Boltzmann Method for Fluid Flows. Annu. Rev. Fluid Mech. **30** (1998), 329–364.
3. http://www.virtualfires.org
4. Filipova, O., Hänel, D. : A Novel Lattice BGK Approach for Low Mach Number Combustion. J. Comp. Phys. **158** (2000), 139–160.
5. Lee, T., Lin, C.-L., Chen, L.-D. : Lattice Boltzmann Simulation of Laminar Jet Diffusion Flame. Proceedings of the 2002 Spring Technical Meeting, Central Sates Section/The Combustion Institute, Knoxville, Tennessee, April 7-9, 2002.
6. Orlandi, P. : Fluid Flow Phenomena. Moreau, R. (ed.) : Fluid Mechanics and its Applications, Kluwer (2000), pp 356.
7. Brandstätter, W., Redl, C. : A LBGK Method for Buoyant Turbulent Combustion Processes. In preparation (2003).
8. Xue, H., Ho, J. C., Cheng, Y. M. : Comparison of different combustion models in enclosure fire simulation. Fire Safety Journal **36** (2001), 37–54.
9. Centre d'Etudes des Tunnels : Les Etudes Specifiques des Dangers pour les Tunnels du Resau Routier. Report Ministere de l'Equipement, des Transports et du Logement (2001), 98 pp.
10. Karki, K. C., Patankar, S. V., Rosenbluth, E. M., Levy, S. S. : CFD Model for Jet Fan Ventilation Systems. Proceedings of 10th International Sysposium on Aerodynamics and Ventilation of Vehicle Tunnels, Boston, November 1–3, 2000.
11. Jones, W. W., Forney, G. P., Peacock, R. D., Reneke, P. A. : A Technical Reference for CFAST : An Engineering Tool for Estimating Fire and Smoke Transport. Report NIST TN **1431** (2000).
12. Formaggia, L., Gerbeau, J. F., Nobile, F., Quarteroni, A. : On the Coupling of 3D and 1D Navier-Stokes Equations for Flow Problems in Compilant Vessels. Comp. Meth. Appl. Mech. Eng. **1991** (2001), 561–582.
13. http://www.vircinity.com
14. http://www.pdc.kth.se
15. Pucher, K. : Konzepte der Tunnel-Brandrauchentlüftung. Längs-, Quer- oder Vollquerlüftung? Proceedings Austroschutz 99, Österreich, 1999.
16. Brandstätter, W., Mawa-Isaac, E., Redl, C. : CFD Simulations of Fire Hazards in the Mont Blanc and Gleinalm Tunnel. VIRTUALFIRES Project Report 5.1, (2002).

Rendering Large (Volume) Datasets: A new Parallel Visualization System

Sascha Schneider
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
sschneid@igd.fhg.de

Thorsten May
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
tmay@igd.fhg.de

Michael Schmidt
Fraunhofer IGD
Fraunhoferstr. 5
64283 Darmstadt, Germany
mschmidt@igd.fhg.de

ABSTRACT

In this paper we describe a basis for a system that is able to compute actual scientific and realistic visualization methods in parallel. It is capable to integrate easily in modern VR renderers like for example *Open Inventor* [SGIa], *Coin* [Coi] and *OpenSG* [Vos02]. Our approach is advantageous for processing large datasets which usually are the result of physically based simulation algorithms and programs. Using our techniques it is even more feasible to manage similar visualization problems for other large amounts of data (e.g. medicinal CT-scans or large geometries) in the context of displaying interactively.

Keywords

Interactive, Parallel Visualization, Computational Fluid Dynamics, Large Volume Data, Visualization System

1. INTRODUCTION

Besides of the simulation of physical phenomena their performant and professional visualization comes more and more into the focus of modern scientific applications. Nowadays there are several powerful physical based simulation programs available on the software market (e.g. *Fluent* [Flu], *FemLab* [Fem], *Flovent* [Flo], *CFX* [CFX], etc.) which allow the user and developer to simulate and investigate nearly every kind of physical problem in high quality and detail. In the realisation concepts of these products mostly parallel approaches play an important role in the program architecture to gain major increases in perfomance.

Empirically these simulation programs are producing large amounts of result data which are often displayed only roughly or inperformant using simple visualization tools. These tools are mostly already integrated within the simulation programs themselves. There are only few visualization programs available which are completely independent of the underlying simulation system and/or data format, grid and glyph types (e.g. *VTK* [Mar96]). These independent tools offer a good general approach for the visualization problem. On the other hand they often lack in performance to process the large amounts of simulation data in reasonable timeframes.

Very large data sets ($> 256^3$ grid points) mainly cause problems due to the

- data is too large to load into main storage completely
- loading data in the hierarchically ordered memory (hard disk, cache, main memory) takes too much time for qualitative, quantitative and interactive rendering.
- costly calculations for some visualization methods.

For these reasons, techniques from the areas of

- data compression (e.g. wavelet methods)
- parallelization of program code (e.g. multi threading, OpenMP [OMP], MPI [MPI])
- hardware accelerated algorithms (e.g. 3D texturing)
- efficient algorithms / software design (e.g. object oriented programming)
- utilization of efficient software development tools (e.g. C/C++) and libraries (e.g. OpenSG [Vos02], OpenGL [SGIb], Qt [Tro])
- development of portable, functional, easy usable and extendible software

were developed and steadily enhanced up to this day.

2. RELATED WORK

There are many works ([Moo96], [Bar99], [Fre99], [Rus00]) in computer graphics that use methods from some of the above areas. But to our knowledge, there is no comparable and published work that covers all of them. On the contrary, our approach to this topic applies most advanced methods from all of these areas in order to create a visualization tool for very large scientific data that features high rendering quality in real time, a very good functionality and an easy handling.

On the area of displaying simulation results many visualization methods have established today, like for example iso-surfaces, stream- and time-lines and volume rendering. But compared to the massive parallel simulation program, the corresponding available visualization software is still noticable less performant. The reason for that is that it mostly works either sequentially or it is implemented as a post process which operates on the basis of an offline rendering concept.

The transfer of the principle of parallel programming from the simulation of physical problems to their interactive visualization lets the end user profit of significant accelerations. This approach enables the applications to process even larger amounts of data interactively. Interactive Visualization of scientific data is a classical area of computer graphics that is steadily and rapidly developing due to the increasing requirements on data volume, display quality, program functionality and usability.

Approaches to interactive visualization in the past partly based on completely (data pre-processing and rendering) parallelization of existing visualization algorithms. By utilizing modern graphics hardware (NVIDIA GeForce [NVI], ATI Radeon [ATI]) it's possible to shift the rendering process to them. The advantage is a faster rendering on standard PCs, that do not provide many processors for parallel execution.

3. ARCHITECTURE

The main target of the following paper is to present a general concept for an interactive parallel visualization of scientific simulation data making use of the sophisticated capabilities of modern graphic cards. In this concept a generalized description of visualization methods is defined. Based on this description an extension of the developed system with further visualization methods is easily realizable every time. For that reason the system can be supplemented rapidly with new visualization methods as soon as they appear.

Furthermore our approach is realized platform independently, to satisfy the need for software portability because of different computer system architectures which are available and in use nowadays. The used in-

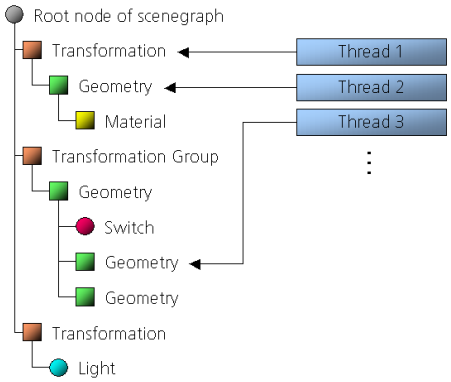

Figure 1: Example scenegraph with multiple threads modifing it.

gredients (C++, OpenGL, OpenSG and QT) allow us to fullfill this capability.

A rather new method in the field of scientific CFD visualization is to render scientific CFD data (stream lines, iso surfaces, ...) combined with detailed and textured geometrical data. This could be, for instance, a 3D model of the original scene that is used from the simulation side for generating the simulation mesh. Blending in original 3D models simplifies navigation for the user and is basis for another technical innovation, the application of highly realistic visualization methods within the area of CFD visualization. This means that in addition to visualization of abstract physical quantities like temperature, pressure, etc. with scientific visualization algorithms (cutting plane, glyphs, iso surface, ...), realistic quantities like fluid, gas or fire and smoke can be rendered as they appear in their natural form inside a photo realistic rendered virtual (simulation) environment [Sch02].

4. SCENEGRAPHS FUNDAMENTALS

Scenegraphs APIs (e.g. Open Inventor / Coin, OpenSG) are immediate APIs in which the objects and commands that are going to be rendered are not sent directly to the graphics processor (GPU) but are integrated in a (acyclic directed) tree graph based description of the displayed scene. This tree is then traversed and rendered separate from the application that provides the geometry and what shall be rendered. Therefore the scenegraph implements a kind of abstraction layer between the application and the GPU. Typical Objects in a scenegraph are normally derived from a base object ("'scene graph tree node"'): Geometry node, material node, transformation node, group node, light node.

Modern scenegraphs like OpenSG support parallel processing (fig. 1) natively so that it possible for the

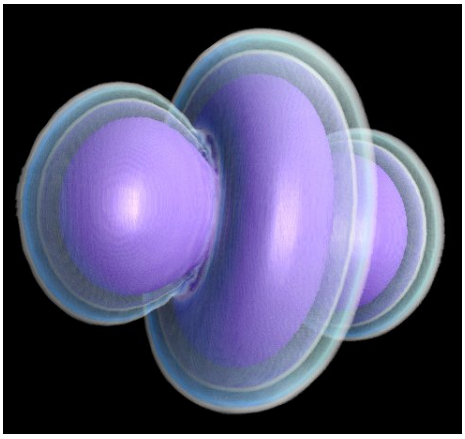

Figure 2: Visualization method: Iso surfaces.

application to modify parts or nodes of the scenegraph directly from several threads at the same time. As long as every application thread addresses a different node in the graph it is not necessary to lock the whole graph each time a part of it is accessed. In spite of this it is possible that every application thread can work exclusively and in parallel on its part of the scenegraph with no risk of creating an access conflict with other nodes / threads. These scenegraphs are called "'thread safe"'.

In this scenegraph we store the geometry of the surrounding (physical) scene (e.g. a car, an airplane or a tunnel) together with the data necessary for rendering the visualization methods which is after all render geometry (triangles, textures, colors, etc.) as well.

5. RENDERING

As mentioned in the beginning nowadays many methods for scientic visualization have established (e.g. iso-surfaces, stream lines, time lines, etc. - (see fig. 2, 3 and 4))

As every 3D rendering, these methods can be created in two ways: through ray tracing/casting on the one hand and through direct rendering using the capabilities of modern graphic cards (e.g. vertex and pixel shaders, shadowing, ...). To provide a parallel approach for the second method it is advisable to make use of thread safe scenegraphs in the first place. By doing so the application can process the calculation for each visualization method in parallel. After the calculation has finished each thread can store its calculated render information in a separate node in the scenegraph. Therefore it is easy to have several visualization methods at the same time in one displayed scene and to process each method in parallel. Furthermore each visualization method itself can be computed in parallel. By doing so, it only has to be assured, that the threaded calculations of one parallelized method itself

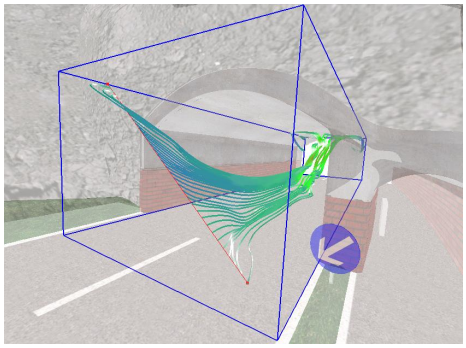

Figure 3: Visualization method: Stream lines.

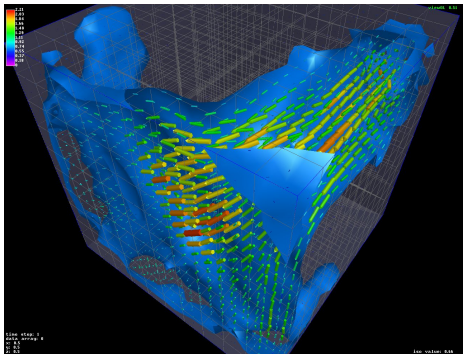

Figure 4: Visualization methods combined: iso-surface and vectors.

is brought together at the same time into the scenegraph to avoid flickering effects.

Probes

To allow the user to restrict visualization methods to certain areas of the datafield/scene we implemented the probe concept [Bar99]. The user can create as many of these probes (fig. 5) and place them in the scene as he wants. Each probe is associated with a cube in VR. In this cube only one certain visualization method is calculated. These probes can be placed arbitrarily and resized within the scene so that it is possible to let the system render the desired visualization method at every place in the dataset where the user wants it to be. To have two or more visualization methods displayed at the same location at the same time it is only necessary to place the corresponding probes together at the same coordinates in VR space.

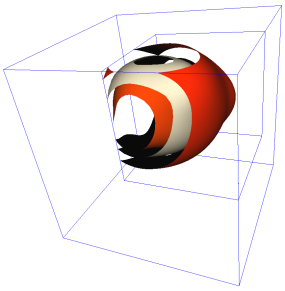

Figure 5: Iso-surfave probe (small cube) placed in the scene (big cube).

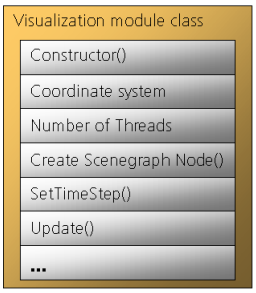

Figure 6: Basic functions and attributes of the general visualization module class.

Class Hierarchy

To allow an easy extension of the system afterwards we implemented a basic class of a visualization method. Based on this generalized description we implemented all visualization methods we needed. If the user wants to introduce a new (specialised) visualization method to the system, the only thing he has to do, is to inherit from the base (= *"'root"'*) class and implement its basic functions (calculation, node generation function, ...). Afterwards he only has to make sure to register his new visualization method within the system. Then he can start right off using his new method.

As shown in fig. 6 the basic functions of the general visualization module class consist of a class constructor, a coordinate system, the number of threads which will be used for calculations, a function to create a scenegraph node for the corresponding VR System, a function for setting the timestep which is used and a function to update the node = (re-)start the calculations. Now every new visualization method which is introduced to the visualization system has to inherit from this class and implement the derived functions according to their functionality.

Additionally each visualization method has to provide a user interface ("'panel-window"') (derived again from a basic generalized class) to the system together with a list of corresponding actions / commands. Every time a feature of an activated visualization method is changed (e.g. the user moves a slider in the interface for the color distribution of a certain method) an action transporting the changed parameters is sent through a inter-thread communication framework (by generating events which are collected in queues) to the central scheduler (see following section). This scheduler now interprets the upcoming events and assigns them to the corresponding active visualization modules. The interpretration of the incoming actions and events is implemented in the visualization module classes and not in the scheduler. This encapsulation allows the scheduler to be as general and flexible as possible. Furthermore it is of course possible to generate these kind of parameter changes not just by hand but by automated control over time for example.

6. PARALLEL CONCEPT

In this section the basic system layout (fig. 7) of the parallel visualization system is introduced showing how the different central parts of the software work together. The system is designed so that it is capable to check the features of the hardware platform it is running on (e.g. count the number of available CPUs or the amount of memory). Each visualization method is implemented fully scalable so that is possible to adjust the number of used calculation threads automatically by the system.

System Design

As one can see in fig. 7 user interacts with the scene and changes the parameters of the visualization. He has influence on the view of the scene (i.e. which part of the scene is rendered from which perspective). Additionally he can create and modify probes each carrying one certain visualization method. On the other side we have the kernel - we call it *"'central scheduler"'*, which collects all incoming actions / events and processes them (see fig. 8).

Inheritance

As mentioned in section each visualization method is derived from a basic class which introduces all system necessary functions for the calculation (and the rendering) as virtual functions. The calculation part of the visualization method is implemented using multiple threads to support the intended parallel processing of the system. Every time the calculation part is finished an event is generated and sent to the kernel.

Kernel

The central scheduler is responsible for processing all necessary reactions of incoming user and / or calcula-

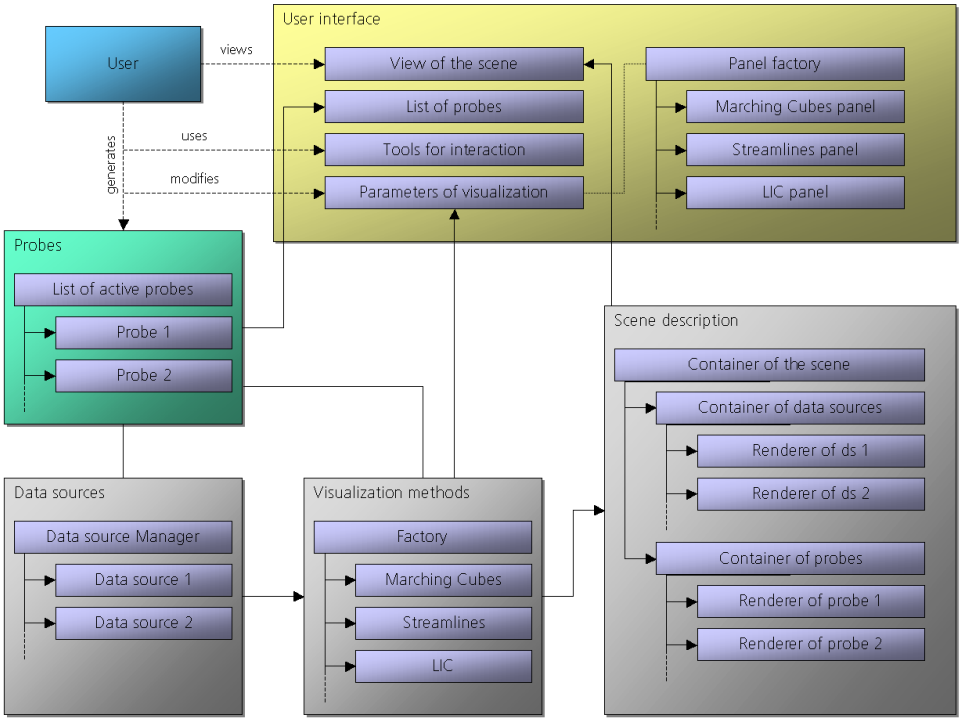

Figure 7: The basic system layout.

tion events. Every time a parameter change is generated, by user interaction or by a timer function modifying it for example, this scheduler receives a corresponding event in his incoming event queue. Being a normal thread it is then activated by the operating system having a look at his *"'incoming queue"'*. Afterwards the scheduler is responsible for initiating and controlling the necessary calculations triggered by the corresponding event. At the end the calculation-threads inform the central scheduler, again using inter-thread event communication, that the calculations have been finished. The scheduler then initiates a scenegraph node generation of the corresponding visualization method and controls the exchange of the old node(s) in the current scene description with the new one(s).

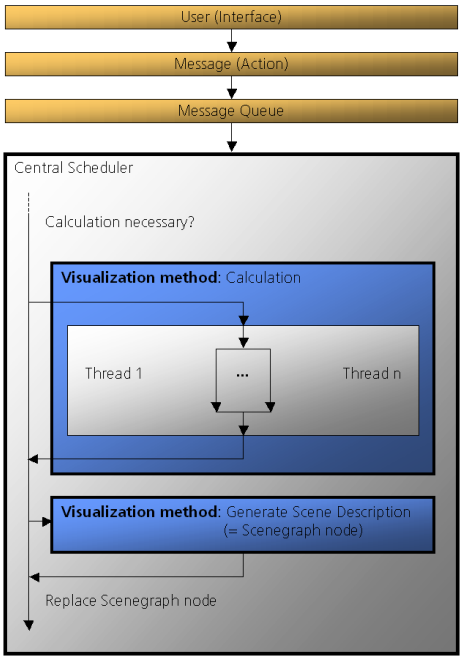

Figure 8: The central scheduler.

7. DATA REPRESENTATION AND STORAGE

In order to adapt the data source management to the resources of the system used, we developed data structures which allow us to trade off accuracy, quality and rendering performance: the *progressive grids*. The progressive grids are a special kind of *hierarchically structured grids* [Wil92], which originate from the field of computer graphics. There they are used to spatially arrange large amounts of geometry data. Progressive grids make use of the technology

of progressive data formats which are closely related to digital image processing (e.g. *Wavelet-, JPEG-Compression*). In these format the data is ordered by its level of detail. So the data associated with a given arbitrary resolution can be extracted efficiently. As a result the amount of data that has to be transmitted and/or processed can be freely adapted to variable system resources or user requirements. Geometry data, for example, can be transmitted and displayed simultanously in incremental granularity levels thus the viewer gets an impression of the geometry already with beginning of the transmission [Hop96]. While using progressive data formats in the context of CFD simulation data, we take advantage of these properties. It makes sense to arrange the data with respect to the information it contains. According to this alignment we built a hierarchy consisting of a spatial partitioning scheme [Sam84] that has so far been used in computer graphics or geometry.

With our work we introduce the principle of progressive data processing to CFD-data. Actually no grid used in numerical simulation is able to handle its data progressively. So these grids have to be converted into the progressive format to make use of them in our visualization system. For this conversion we are free to choose which cell types, partitioning schemes or error estimation schemes are used in concrete. All these parameters can be selected independently from each other and this offers a rich repertoire of possibilities. According to this, the converter is divided in two parts: A fixed one and a plug-in, which manages cell type information, its topology, interpolation schemes etc. The plug-in part can be replaced in order to implement different progressive grids (i.e. grids which use different cell types, partitioning schemes etc.).

The conversion itself works in the following way: The bounding box of the original simulation grid becomes the *root cell* of our progressive grid. Using the predefined decomposition scheme, a hierarchy of grid cells is then built up through spatial partitioning. An approximation error is computed through comparision of values interpolated within the current cell and the original grid. (This approximation is independent from the topology of the original grid.) A cell will be further partitioned, if its approximation error is the worst compared to all other currently unpartitioned cells (see fig. 9). The new cells generated in this way, represent a *"'better"'* approximation of the original grid. The whole partitioning process stops if a certain error tolerance has been reached.

The progressive grid generated in this way has a number of advantages. The maximum error within the leaf-cells in the hierarchy decreases fast. This minimizes the number of cells to be loaded at a given error

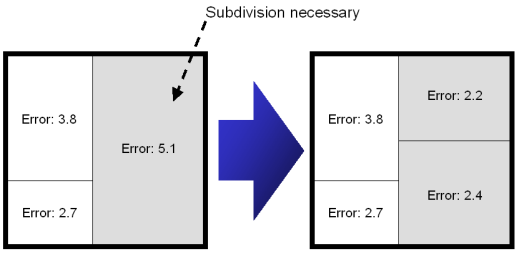

Figure 9: Subdivision: A cell will be partitioned if the error is worse compared to the error of the other cells.

tolerance. Furthermore these cells constitute a single block, because they are written in the same order their parent cells have been partitioned. The hierarchical structure of the grid can be exploited for compression in areas where low-frequency portions of the scalar fields predominate. We converted a number of datasets of fire simulations and are able to zoom through the levels of detail in real time, without making concession to performance. A visualization of streamlines (involving 200.000 vertices) using the progressive grid was comparable in speed to the one on the original, equidistant grid.

8. SUMMARY

We presented a new concept of a visualization system which is able to make use of the capabilities of modern graphic cards. This system is scalable and can be easily adjusted to different hardware conditions. Furthermore it is portable and can be easily extended with new visualization methods. Together with its capability to display scientific visualization methods together with realistic rendered it is very attractive to the end user, because he is able to investigate his simualtion results within a realistic virtual environment. The parallel approach of our system makes it very attractive for processing very large amounts of (simulation) data. Based on the progressive approach it becomes possible to visualise even large datasets on machines which have only little performance only at the cost of losing details in the loaded and displayed data.

9. REFERENCES

[Tro] Trolltech AS. Qt, c++ toolkit for application development. http://www.trolltech.com/products/qt/.

[Moo96] Robert Moorhead Aaron Trott and John McGinley. Wavelets applied to lossless compression and progressive transmission of floating point data in 3-d curvilinear grids. In *Proceedings IEEE Visualization '96*, pages 385–388. IEEE, 1996.

[OMP] OpenMP Architecture Review Board. Openmp - simple, portable, scalable smp programming. http://www.openmp.org/.

[CFX] CFX. Cfx, cfd software package. http://www.software.aeat.com/cfx/.

[Coi] Coin3D. Coin, scenegraph based 3d graphics library. http://www.coin3d.org/.

[Vos02] Gerrit VoßDirk Reiners and Johannes Behr. Opensg - basic concepts. http://www.opensg.org/OpenSGPLUS/symposium/Papers2002/.

[Fem] Femlab. Femlab, pde solver package. http://www.femlab.com/femlab/.

[Fre99] Lori A. Freitag and Raymond M. Loy. Adaptive, multiresolution visualization of large data sets using a distributed memory octree. In *Proceedings of SC99: High Performance Networking and Computing*, 1999.

[Hop96] H. Hoppe. Progressive meshes. In *SIGGRAPH '96: Proceedings*, 1996.

[ATI] ATI Techonogies Inc. Ati radeon, graphic board. http://www.ati.com/.

[Flu] Fluent Inc. Fluent, cfd software package. http://www.fluent.com/software/fluent/.

[Flo] Flomerics Ltd. Flovent, cfd based software. http://www.flovent.com/.

[MPI] mpi forum.org. Mpi - message passing interface. http://www-unix.mcs.anl.gov/mpi/indexold.html.

[Bar99] W. Bartelheimer M. Schulz, F. Reck and T. Ertl. Interactive visualization of fluid dynamics simulations in locally refined cartesian grids. In *Proceedings IEEE Visualization '99*, pages 413–553. IEEE, 1999.

[NVI] NVIDIA. Geforce, graphic processing unit. http://www.nvidia.com/.

[Rus00] Szymon Rusinkiewicz and Marc Levoy. Qsplat: A multiresolution point rendering system for large meshes. In *Siggraph 2000: Computer Graphics Proceedings*, 2000.

[Sam84] H. Samet. The quadtree and related hierarchical data structures. *ACM Computing Surveys (CSUR)*, 16(2), June 1984.

[SGIa] SGI. Open inventor, object oriented 3d graphics api. http://www.sgi.com/software/inventor/.

[SGIb] SGI. Opengl, 3d rendering api. http://www.opengl.org/.

[Sch02] Sascha Schneider Thorsten May and Volker Luckas. Parallel real time fluid simulation and animation with fractal optical refinements. In *ESM 02: Proceedings of the 16th European Simulation Multiconference, Modelling and Simulation 2002*, pages 224–228, 2002.

[Mar96] Kenneth M. Martin William J. Schroeder and William E. Lorensen. The design and implementation of an object-oriented toolkit for 3d graphics and visualization. *IEEE Visualization '96*, http://public.kitware.com/VTK/:93–100, 1996.

[Wil92] J. Wilhelms and A. van Gelder. Octrees for faster isosurface generation. *ACM Transactions on Graphics (TOG)*, 11(3), 1992.